(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 321 089 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 88310526.4

(51) Int. Cl.4: G06F 5/06

(22) Date of filing: 09.11.88

(30) Priority: 18.12.87 US 135170

(43) Date of publication of application:
21.06.89 Bulletin 89/25

(84) Designated Contracting States:
DE FR GB

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Lee, Chien-Chyun
1302 Sawgrass Cove
Austin Texas 78746(US)
Inventor: Moore, Charles Roberts
2105 Margalene Way
Austin Texas 78728(US)

(74) Representative: Hawkins, Anthony George Frederick
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

(54) Digital data buffer storage system.

(57) A buffer for storing data words consisting of several storage locations together with circuitry providing a first indicator that designates the next storage location to be stored into, a second indicator designating the next storage location to be retrieved from, and circuitry that provides the number of locations available for storage and the number of locations available for retrieval. The buffer includes the capability to store and retrieve several data words simultaneously.

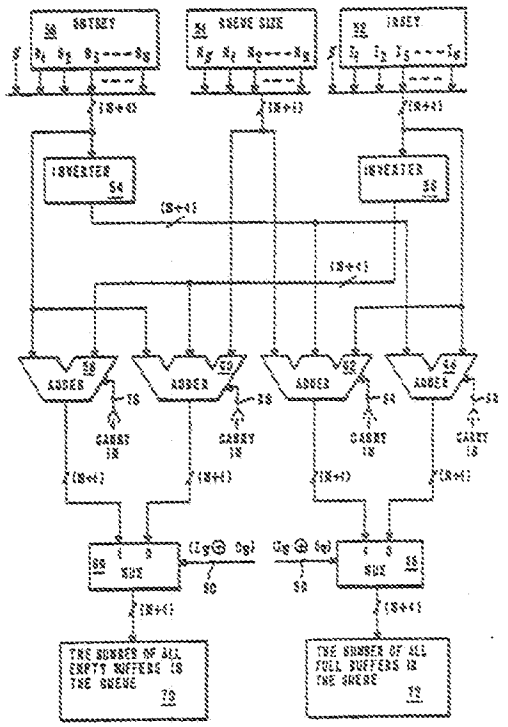

FIG. 2

EP 0 321 089 A2

DIGITAL DATA BUFFER STORAGE SYSTEM

The present invention relates to data processing systems, and in particular to buffers for temporarily storing data.

Data processing systems perform their functions by moving data and performing operations on this data. Often it it necessary to temporarily store data that is being transferred from one device to another in a data processing system. Circuitry for this temporarily storage is often called a buffer. Commonly, buffers are use to store several data words at a time.

Buffer management requires that the location of data stored in the buffer be known. Specifically the next available location for storage and the location previously stored must be known. Furthermore, there must be a capability for determining when the buffer is full or empty.

One type of buffer that is quite commonly used in data processing systems is termed a circular buffer or circular queue. In a circular queue, there is no bottom or top to the queue but the queue positions are circular in arrangement. A circular queue may contain any number of storage positions. The management of the circular queue requires that the "head" and the "tail" of the queue must be known. The head is the next position in the queue that is to be written into. The tail is the next position of the queue that is to be read from. The tail of the queue must always follow the head of the queue. However, the head of the queue should never lap the tail of the queue.

In general, the head of the queue is pointed to by an "inkey" pointer. The tail of the queue is pointed to by an "outkey" pointer. A capability must be provided to insure that the inkey pointer does not lap the outkey pointer.

U. S. Patent 3,771,142 entitled "Digital Data Storage System" discloses a circular queue that is used to store data for several input/output devices.

IBM Technical Disclosure Bulletin, volume 20, number 8, January, 1978, pages 3309-3310, entitled "Data Management in a Circular Buffer", discloses a technique to manage the operation of a circular buffer. IBM Technical Disclosure Bulletin, volume 24, number 12, May, 1982, pages 6240-6243, entitled "Modular Buffer Allocation Control Logic", discloses circuitry for controlling several data buffers that are being written to and read from in sequence. The circuitry illustrated provides an indication of when buffers are available for read and write operations. IBM Technical Disclosure Bulletin, volume 20, number 10, March, 1978, page 4130, entitled,"FIFO Queue Edit Mechanism", discloses a technique for preventing the head pointer from wrapping the tail pointer.

A current trend in data processing is that of parallel processing. In parallel processing several data operations are conducted simultaneously. Such an environment places unusual demands on other elements in the data processing system, such as data buffers or data queues.

It is an object of the present invention to provide a dynamic buffer that is capable of performing a multiple of data reads or data writes in a single machine cycle.

According to the invention, there is provided a digital data buffer storage system having a plurality of storage locations, means providing a first indicator designating the next storage location available to be written into, means providing a second indicator designating the next location to be read from, and means coupled to the buffer store, said first means and said second means, for controlling simultaneous read from or write to a plurality of storage locations available for said read or write as defined by the first and second indicators.

In the preferred embodiment, the buffer is a circular queue. The circular queue includes an inkey pointer for indicating the next storage location to be stored into and an outkey pointer for indicating the next storage location to be retrieved from. Further control circuitry is provided that includes the means to simultaneously compute from the inkey and outkey pointers and the queue size, the number of empty buffer locations and the number of full buffer locations. These computations are used by the control circuit to provide simultaneous storage or retrieval of several data words.

The preferred embodiment will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a data buffer;

Figure 2 is a block diagram of buffer control circuitry that provides simultaneously the number of buffer positions available for storage and the number of buffer locations filled;

Figure 3 is a flowchart representing the computational flow of the embodiment in Figure 2;

Figure 4 is a second embodiment of the buffer control circuitry that computes the number of available and the number of filled buffer locations;

Figure 5 is a flowchart illustrating the computational flow for the circuitry in Figure 4;

Figure 6 is a third embodiment of the buffer control circuitry wherein the number of buffer locations is a power of 2; and

Figure 7 is a flowchart illustrating the computational flow for the circuitry in Figure 6.

The present system relates to a buffer that provides a temporary storage of data words. One uniqueness of the present buffer is that it provides for the simultaneous storage or retrieval of several data words during a single processing cycle. This feature is very important for parallel processing systems. This enables the parallel processing systems to store or retrieve simultaneously data words that are being used in the different data processing operations that are being conducted simultaneously.

Figure 1 is a block diagram that illustrates the embodiment of the present invention. In Figure 1, and instruction cache 10 provides the temporarily storage of a large number of individual instructions for execution by a data processor. These instructions are treated by the buffer as data words. The instruction cache 10 provides an output over line 22 to a multiplexor 16. The purpose of the multiplexor 16 is to receive simultaneously from the instruction cache 10 several instructions or data words which are then specifically aligned by an inkey pointer and the available storage in buffer 20 for parallel transfer across lines 24 into the buffer 20. A buffer control circuit 12 provides the control signals on line 26 to the instruction cache 10 to initiate the information transfer across lines 22. Line 9 includes control signals from the buffer control 12 to multiplexor 16 to align the instructions in accordance with the inkey and the available storage in buffer 20. The buffer control 12 further controls storage of information on the parallel lines 24 to buffer 20 with control signals transmitted to buffer 20 on line 28. Buffer 20 provides an output of several data words simultaneously on line 30 to a multiplexor 18. Multiplexor 18 maintains the order of the instructions in the buffer. In other words, the instructions that were first read into the buffer 20 are first read out. Buffer control 12 provides control signals to multiplexor 18 over line 17 to regulate the sequential ordering according to an outkey and the number of instructions available. It should be apparent to those skilled in the art that this multiplexor 18 may be rearranged to operate in a last in/first out manner. The output from multiplexor 18 is provided on lines 32 to receiving logic 22. In the preferred embodiment, the instructions that are output from buffer 20 are provided to instruction interlock logic (i.e. block 22) for further preprocessing before execution.

The buffer control circuit 12 operates in a manner common to most circular buffer controllers. In other words, the buffer control circuit 12 includes an inkey pointer pointing to the first buffer location in buffer 20 that is available for storage of a data word. Buffer control circuitry 12 further includes an outkey pointer that points to the first available buffer location in buffer 20 that contains data to be read from buffer 20.

Circuitry 14 of buffer control circuitry 12 computes the number of empty buffer location in buffer 20 and the number of filled buffer locations in buffer 20. These computations are accomplished simultaneously. Further, buffer control circuitry 12 includes additional circuitry to ensure that the inkey and outkey pointers are appropriately incremented and that the inkey and outkey pointers do not wrap around each other.

Figure 2 represents the circuitry in block 14 (Figure 1) that provides the number of empty locations in the queue and the number of full locations in the queue. Referring to Figure 2, latch 50 stores the outkey indicator. Latch 52 stores the inkey indicator. Latch 51 stores a number indicating the size of the queue. An inverter 54 is connected to the output of the outkey latch 50. Likewise, an inverter 56 is connected to the output of the inkey latch key 52. The output of inverters 54 and 56 together with the output of latches 50, 51 and 52 are provided to four adders 58, 60, 62 and 64 as indicated to provide 1's complement inputs. The carry input to adders 58, 60, 62, and 64 (i.e., carry in lines 76, 80, 84 and 88 respectively) are set to +1 to convert the 1's complement inputs for 2's complement operation. Adder 58 performs the arithmetic computation of subtracting the inkey indicator stored in latch 52 from the outkey indicator stored in latch 50. Adder 60 likewise subtracts the inkey indicator in latch 52 from the outkey indicator in latch 50 and adds in the queue size from latch 51. Adder 62 subtracts the outkey indicator from latch 50 from the inkey indicator in latch 52 and adds in the queue size from latch 51. Adder 64 subtracts the outkey indicator from latch 50 from the inkey indicator from latch 52. The output from adder 58 and 60 are input to a multiplexor 66 which provides either the output from adder 58 or the output from adder 60 to latch a 70. The contents of a latch 70 represents the number of empty locations in the queue. Multiplexor 66 is controlled by line 90 which provides a signal indicating the output of an Exclusive OR operation between the inkey wrap indicator and the outkey wrap indicator. Likewise multiplexor 68 is connected to adders 62 to and 64 and provides the output from either adders 62 or 64 to latch 72. The content of latch 72 represents the number of full locations in the queue. Multiplexor 68 is also controlled by line 90.

The wrap indicators for the outkey and the inkey merely represent the relative locations of the pointers with respect to each other. In other words, since a non-circular address sequence is used by the pointers, the wrap indicator indicates whether one indicator is wrapped around the queue relative to the other indicator. In the normal operation the inkey would be incremented before the outkey is

incremented. Except when the queue is empty, the inkey indicator would be numerically greater than the outkey indicator. However, since the queue is operating in a circular fashion, it is possible that the inkey indicator could have "wrapped around" the queue and actually be less than the outkey indicator. In such a circumstance, it would appear that the inkey indicator was incorrect. The wrap indicator for both the inkey and the outkey pointers merely provides a means to determine that such inconsistency has not occurred.

For an example, initially both the inkey and the outkey pointers are 0. The queue is empty. For a queue of size N, and if nothing is removed from the queue, the inkey will be incremented as entries are made. Eventually, the inkey will be incremented beyond N-1 and back to 0 where the outkey has remained. However, the wrap indicator for the inkey will be set to 1. The outkey wrap indicator remains 0. This condition indicates that the queue is full and the queue cannot accept new entries until some are removed. As entries are removed from the queue, the outkey is incremented by the appropriate amount. If nothing new is added in the queue, the outkey indicator will eventually be incremented to wrap around to 0. The position where the inkey remains. The wrap indicator for the outkey will than be set. Therefore, the wrap key for the inkey indicator and the wrap indicator for the outkey indicator would both be equal, indicating that queue is empty.

The operation of the circuitry in Figure 2 is illustrated in Figure 3 as a flowchart. In block 100 the calculation of adders 58, 60, 62, and 64 are indicated. These calculation are made from the contents of the outkey and inkey latches with the wrap indicators not considered. Therefore, referring to block 100, variables A, B and C are computed. Variable A is the subtraction of the inkey from and outkey performed in adder 58. Variable B is the computation involving the subtraction of the inkey from the outkey and the addition of the queue size performed in adder 60. The Exclusive OR operation results in Variable C. In decision Block 102, if Variable C is equal to 1, the number of all empty locations in the queue are set equal to A in Block 104. If C is equal to zero, this number is said equal to B in Block 106.

Likewise, Block 108 represents the computations of Variables D and E. Variable D is the subtraction of the outkey in the inkey performed in adder 64. Variable E is the subtraction of the outkey in the inkey indicator and the addition of the queue size as performed in adder 62. The Variable C used in decision Block 110 is the same that is computed in Block 100. If C is equal to 1, the number of full buffers is the Variable E in Block 112. If C is equal to 0, the number of full locations is set equal to the D Variable in Block 114.

As an example, consider that the outkey wrap indicator equals 0, the outkey is set equal to 1, the inkey wrap indicator set to 0 and the inkey indicator set to 4 with the queue size of 5 Then A would be equal to minus 3, B will be equal to 2 and C will be 0. Therefore, the number of empty locations will be equal to B or 2. Variables D and E would be equal to 3 and 8 respectfully. Since C is 0, the number of full locations is 3.

A second embodiment of the circuitry used to provide the number of empty locations and the number of full locations is illustrated in Figure 4. In Figure 4 the wrap indicators for the outkey and inkey counters have been illustrated as been separated from the latches from the outkey and inkey pointers. The outkey wrap indicator 120 and the outkey pointer latch 122 provide inputs to adder 144 and through inverter 134 to adder 142. The inkey wrap indicator 128 and the inkey pointer latch 130 provide inputs to adder 142 and through inverter 140 to adder 144. Queue size is stored in latch 124. Note that the most significant bit position of the queue size in latch 124 is inverted by inverter 132. The queue size with the inverter most significant bit is provided to multiplexors 136 and 138. A zero correction value in latch 126 is also provided to multiplexors 136 and 138. Multiplexors 136 and 138 are controlled by line 139 which is the Exclusive OR of the inkey wrap indicator 128 and the outkey wrap indicator 120. The output of multiplexor 136 is provided to adder 142. The output of multiplexor 138 is provided to adder 144. The output of adder of 142 is provided to latch 148 representing the number of full locations in the queue. The output of adder 144 is provided to latch 150 to represent the number of all empty locations in the queue. Note that the most significant bit position of the output from adder 144 is inverted by inverter 146 before it is provided to the latch 150.

Figure 5 represents the computational flow performed by the circuitry in Figure 4. In Block 152, the outkey and inkey variables both include the outkey and the inkey wrap indicators. The computations for the Variable C is computed as before, i.e., the Exclusive OR of the outkey wrap key indicator with the inkey wrap indicator. In decision Step 154, it is determined if C is equal to 1. If not, the correction factor is set equal to the queue size with the queue size most significant bit inverted in Step 158. If C is equal to 1, the correction factor is equal to the contents of latch 126 (Figure 4) which is 0. In Step 160, the Variable A is computed which is equal to the outkey indicator minus the inkey indicator plus the corrections factor. In Step 162 the number of all unoccupied buffer locations are set equal to A with the most significant bit inverted. In Step 164, the correction factor is to be set as in

Step 164. In Step 168, the correction factor is set to 0. In Step 166, correction factor is set equal to the queue size with the queue size most significant bit inverted. Then in Step 170, the number of occupied buffers are said equal to the inkey indicator minus the outkey indicator plus the correction factor.

Figure 6 illustrates a third embodiment wherein the queue size is required to be equal to a number that is a power of 2. Figure 6 is similar to Figure 4 except that the correction factor is not required for the computation.

Figure 7 illustrates they computational flow from the circuitry in Figure 6. In Figure 7, block 178 provides the inkey, outkey (with their respective wrap indicators included) and computes the variable C as before. The Variable A is computed in Block 180 as being the outkey minus the inkey indicators. The most significant bit of A is inverted and is provided in Block 182 as being the number of buffer positions that are empty. The Variable B is computed in Block 184 and is equal to the inkey indicator minus the outkey indicator. This is provided as the number of buffer position that are occupied.

Claims

1. A digital data buffer storage system including a buffer store having a plurality of storage locations, means providing a first indicator designating the next storage location available to be written into, means providing a second indicator designating the next location to be read from, and means coupled to the buffer store, said first means and said second means, for controlling simultaneous read from or write to a plurality of storage locations available for said read or write as defined by the first and second indicators.

2. A storage system according to claim 1 in which the number of said available storage locations is less than the storage locations in the buffer store.

3. A storage system according to claim 1or claim 2 including means for providing an indication of the available storage locations and the number of locations occupied by stored data.

4. A system according to any one of the previous claims in which the indications of the next locations are incremented in a circular manner.

5. A system according to any of the previous claims in which data is stored in the buffer store in a first-in-first-out fashion.

6. A method of storing data words in a buffer store having a plurality of storage locations, comprising the steps of:
providing a first indicator defining the next storage location to be written into;
providing a second indicator defining the next storage location to be read from;
providing a third indication defining the number of storage locations available for storage; and
in response to the first, second and third indications, reading or writing a plurality of data words from or into a plurality of said storage locations simultaneously, provided that, for writing, such locations are available.

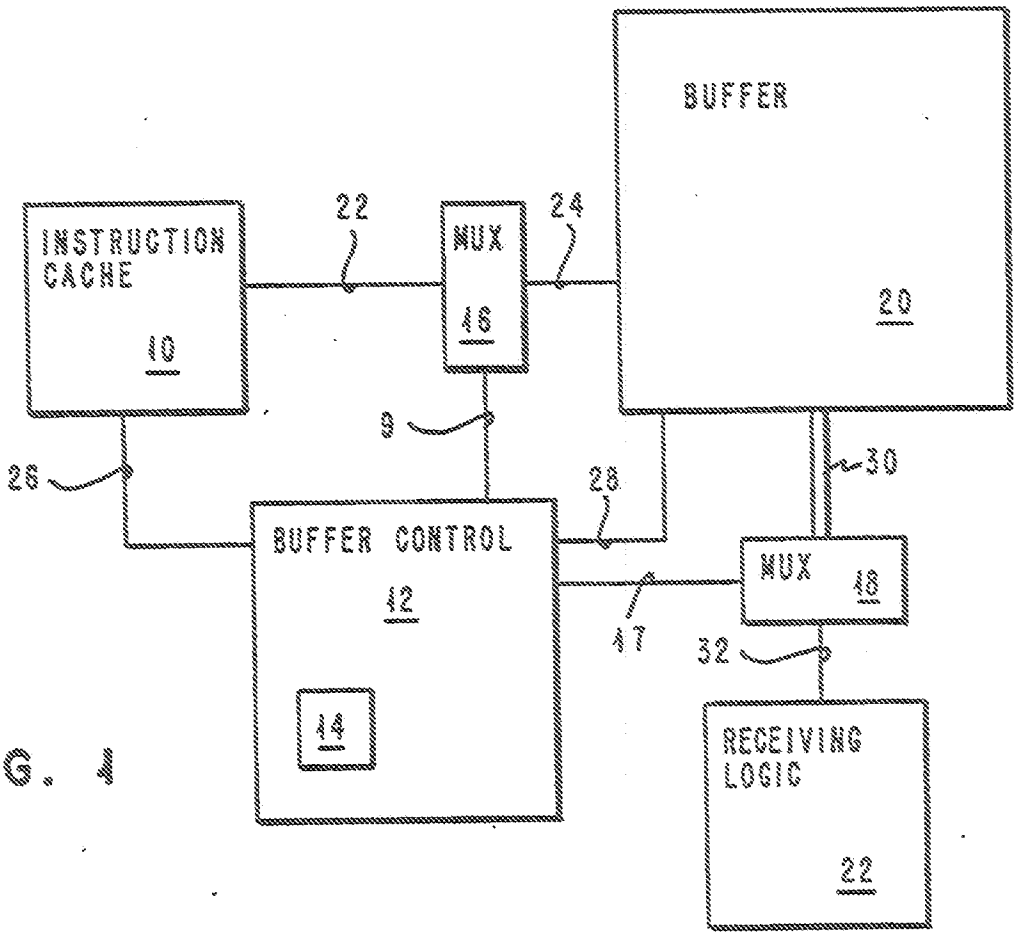
BUFFER
20
INSTRUCTION CACHE
10
22
MUX
16
24
9
26
28
30
BUFFER CONTROL
12
14
MUX
18
17
32
RECEIVING LOGIC
22

FIG. 1

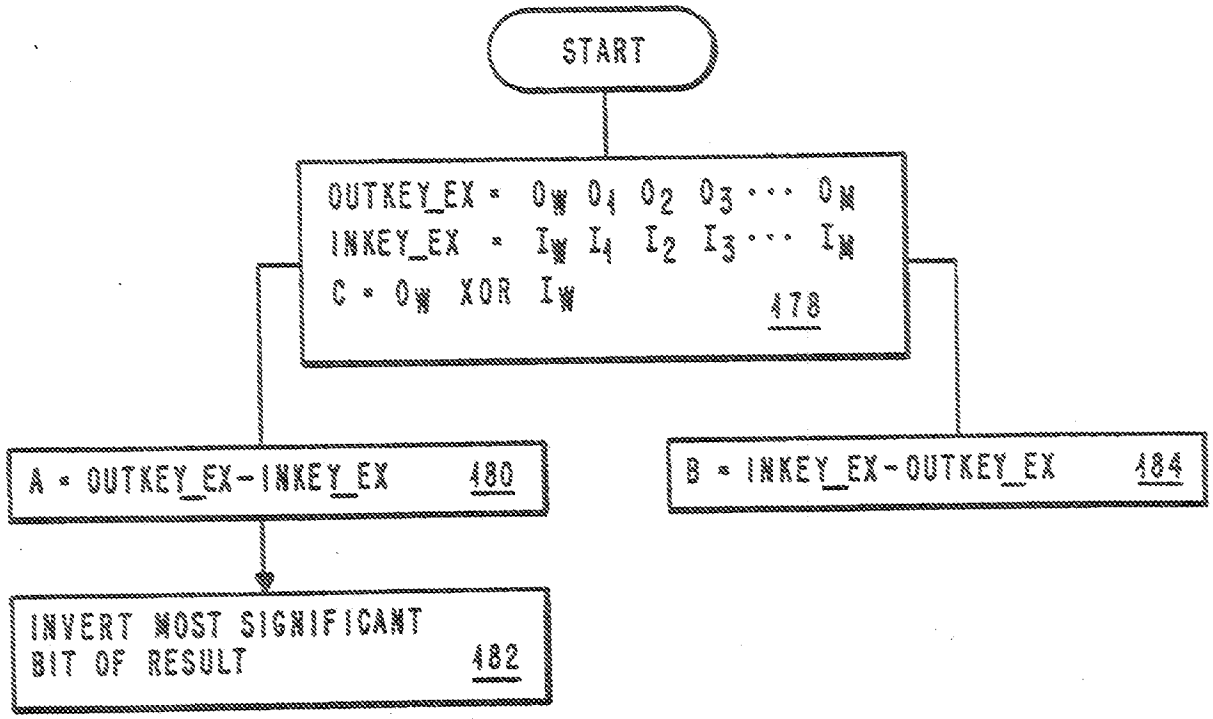
START
OUTKEY_EX = O_W O_1 O_2 O_3 ... O_N
INKEY_EX = I_W I_1 I_2 I_3 ... I_N
C = O_W XOR I_W
178
A = OUTKEY_EX - INKEY_EX
180
INVERT MOST SIGNIFICANT BIT OF RESULT
182
B = INKEY_EX - OUTKEY_EX
184

FIG. 7

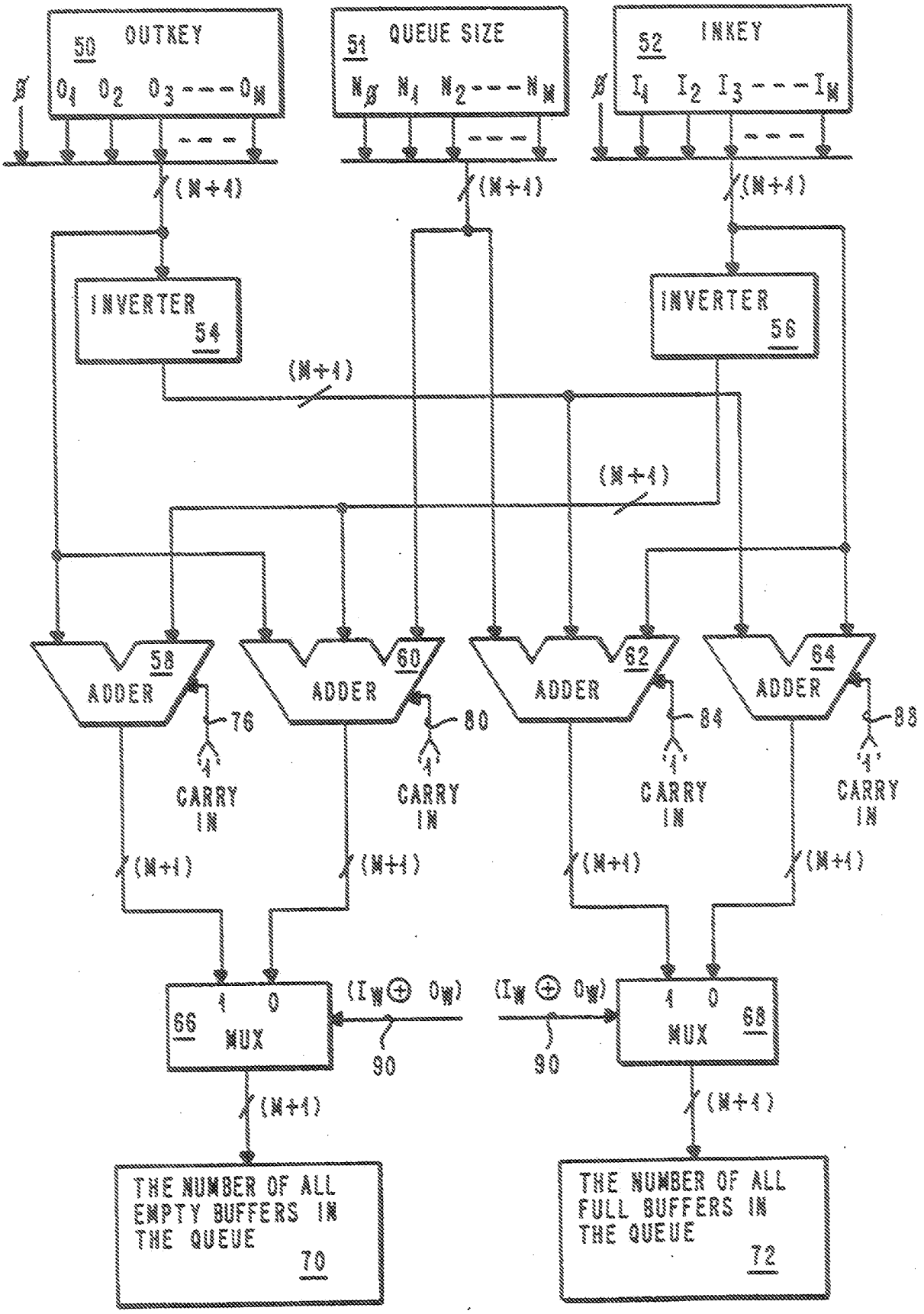
50 OUTKEY
$O_\emptyset$ O_1 O_2 O_3 --- O_N
51 QUEUE SIZE
$N_\emptyset$ N_1 N_2 --- N_N
52 INKEY
$I_\emptyset$ I_1 I_2 I_3 --- I_N
(N+1)
INVERTER 54
INVERTER 56
58 ADDER
60 ADDER
62 ADDER
64 ADDER
76
80
84
88
CARRY IN
66 MUX
68 MUX
$(I_N \oplus O_N)$
90
THE NUMBER OF ALL EMPTY BUFFERS IN THE QUEUE 70
THE NUMBER OF ALL FULL BUFFERS IN THE QUEUE 72

FIG. 2

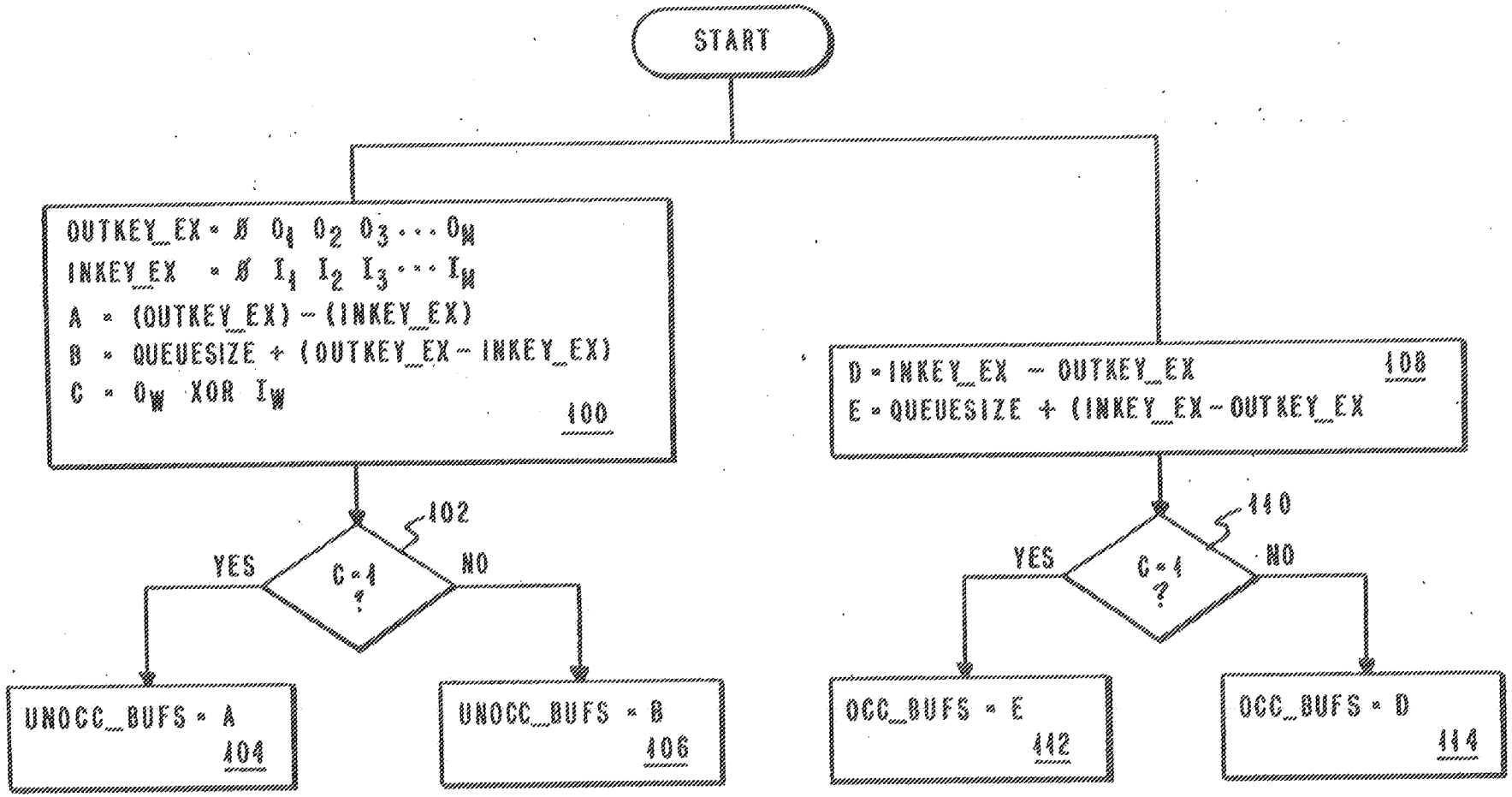
START
OUTKEY_EX = $\emptyset$ O_1 O_2 O_3 ... O_N
INKEY_EX = $\emptyset$ I_1 I_2 I_3 ... I_N
A = (OUTKEY_EX) - (INKEY_EX)
B = QUEUESIZE + (OUTKEY_EX - INKEY_EX)
C = O_N XOR I_N
100
102
C = 1 ?
YES
NO
UNOCC_BUFS = A
104
UNOCC_BUFS = B
106
D = INKEY_EX - OUTKEY_EX
108
E = QUEUESIZE + (INKEY_EX - OUTKEY_EX
110
C = 1 ?
YES
NO
OCC_BUFS = E
112
OCC_BUFS = D
114

FIG. 3

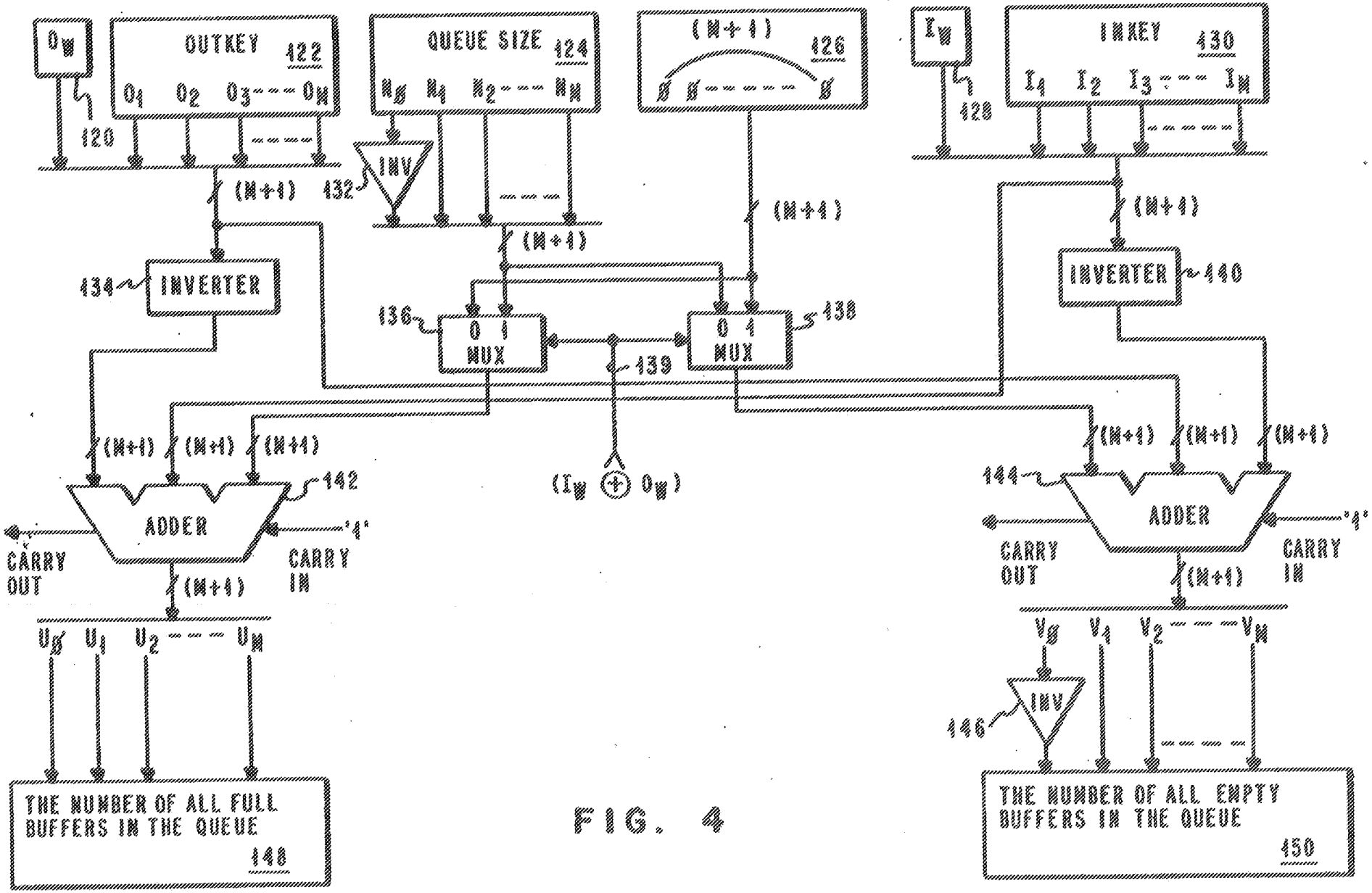
O_W
OUTKEY 122
O_1 O_2 O_3 --- O_M
120
QUEUE SIZE 124
N_0 N_1 N_2 --- N_M
(M+1) 126
0 0 ----- 0
I_W
128
INKEY 130
I_1 I_2 I_3 --- I_M
(M+1)
INV
132
INVERTER
134
INVERTER
140
0 1 MUX
136
0 1 MUX
138
139
$(I_W \oplus O_W)$
ADDER
142
'1'
CARRY OUT
CARRY IN
ADDER
144
'1'
CARRY OUT
CARRY IN
U_0 U_1 U_2 --- U_M
THE NUMBER OF ALL FULL BUFFERS IN THE QUEUE
148
V_0 V_1 V_2 --- V_M
INV
146
THE NUMBER OF ALL EMPTY BUFFERS IN THE QUEUE
150

FIG. 4

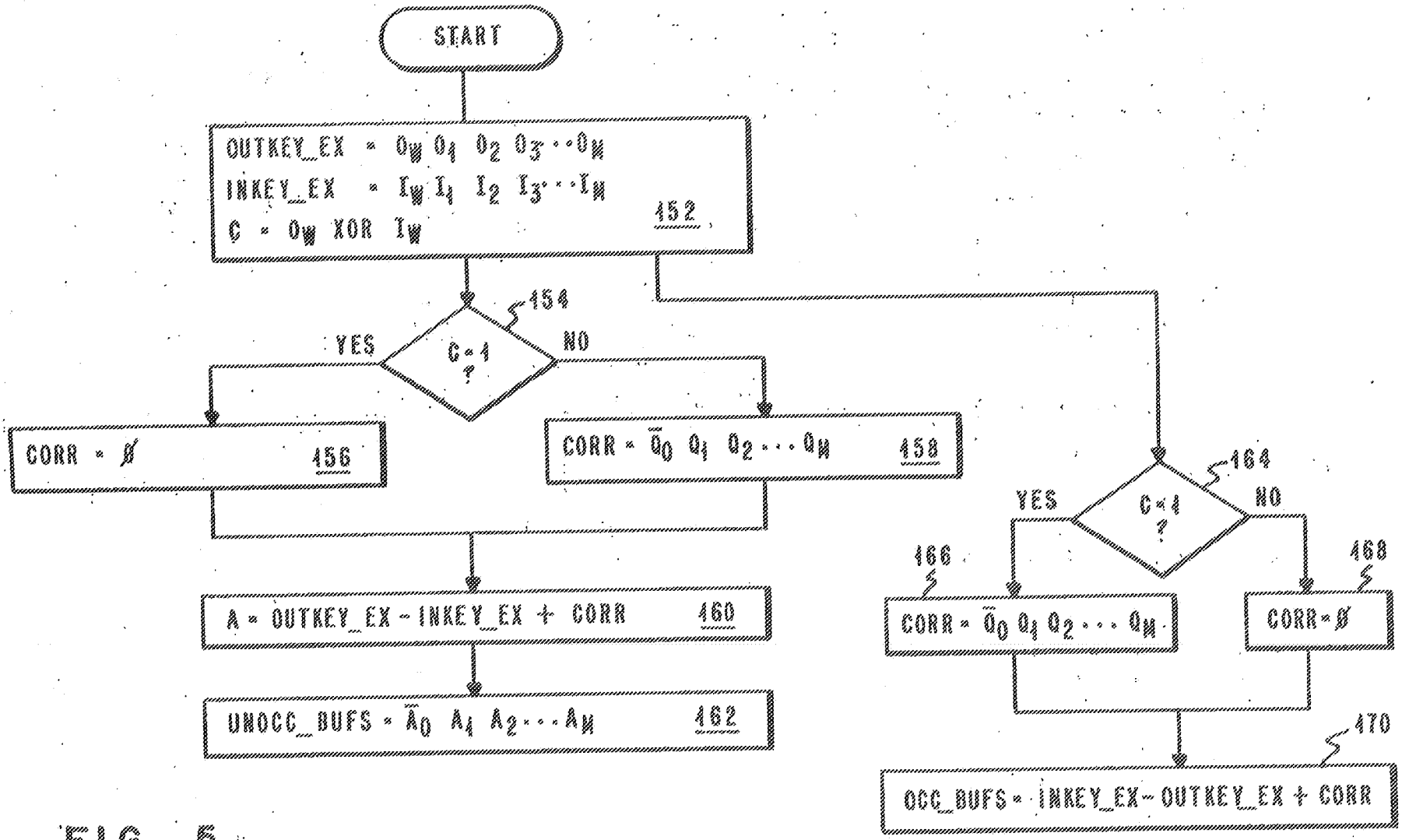

FIG. 5

EP 0 321 089 A2

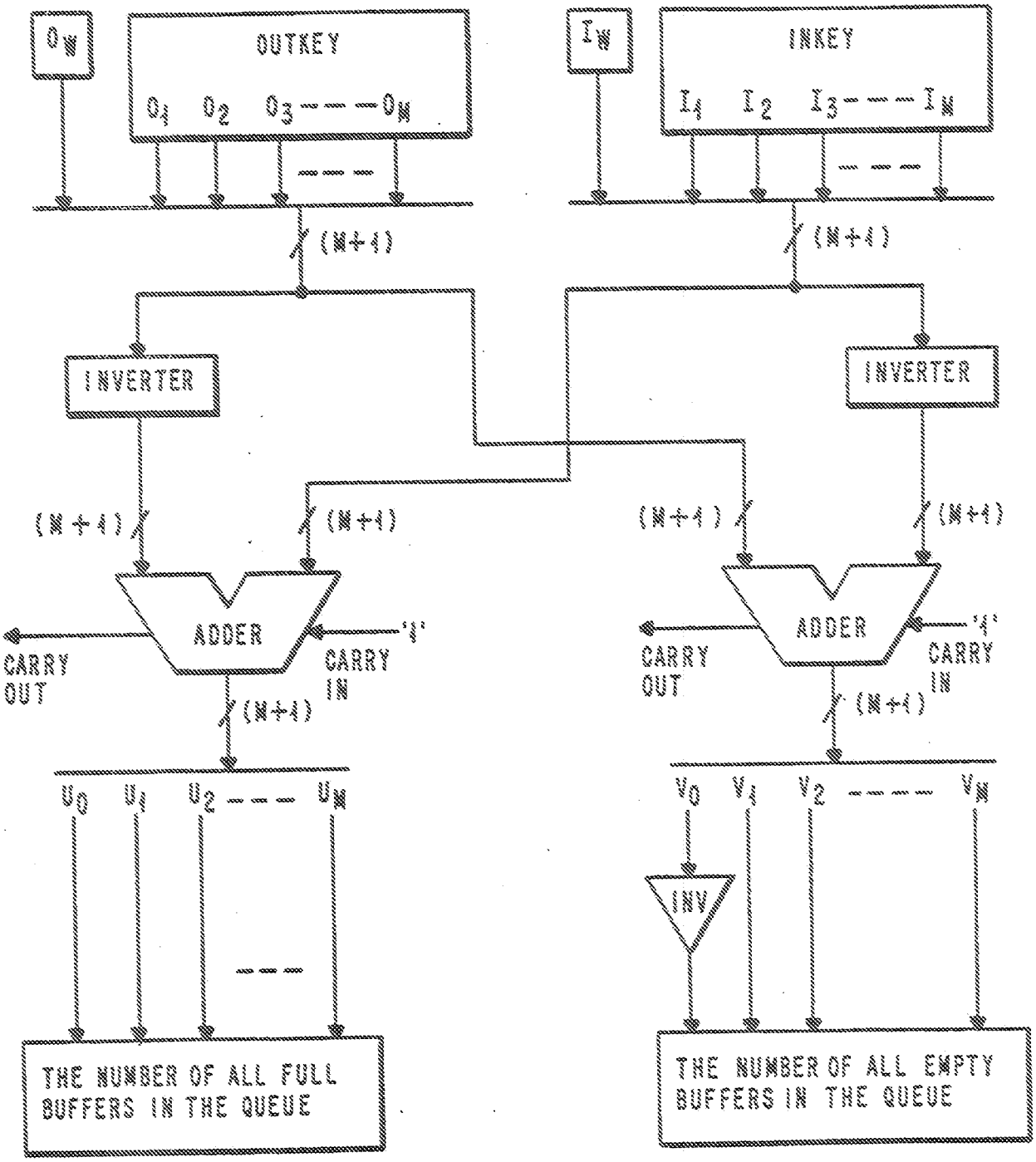
O_W
OUTKEY
O_1 O_2 O_3 - - - O_M
I_W
INKEY
I_1 I_2 I_3 - - - I_M
(M+1)
INVERTER
ADDER
'1'
CARRY IN
CARRY OUT
U_0 U_1 U_2 - - - U_M
V_0 V_1 V_2 - - - V_M
INV
THE NUMBER OF ALL FULL BUFFERS IN THE QUEUE
THE NUMBER OF ALL EMPTY BUFFERS IN THE QUEUE

FIG. 6

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 321 089 A3

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 88310526.4

(51) Int. Cl.5: G06F 5/06

(22) Date of filing: 09.11.88

(30) Priority: 18.12.87 US 135170

(43) Date of publication of application:
21.06.89 Bulletin 89/25

(84) Designated Contracting States:
DE FR GB

(88) Date of deferred publication of the search report:
08.05.91 Bulletin 91/19

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Lee, Chien-Chyun
1302 Sawgrass Cove
Austin Texas 78746(US)
Inventor: Moore, Charles Roberts
2105 Margalene Way
Austin Texas 78728(US)

(74) Representative: Bailey, Geoffrey Alan
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

(54) Digital data buffer storage system.

(57) A buffer for storing data words consisting of several storage locations together with circuitry providing a first indicator that designates the next storage location to be stored into, a second indicator designating the next storage location to be retrieved from, and circuitry that provides the number of locations available for storage and the number of locations available for retrieval. The buffer includes the capability to store and retrieve several data words simultaneously.

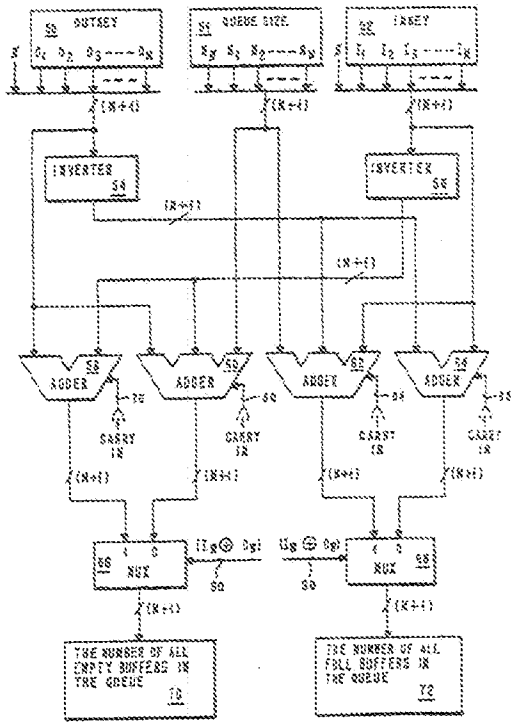

FIG. 2

EP 0 321 089 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 88 31 0526

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	ELECTRONIC DESIGN, vol. 35, no. 14, 11th June 1987, pages 117-122, Hasbrouck Heights, NJ, US; T. OLSON: "Variable-width FIFO buffer sequences large data words" * Page 117, column 1, line 1 - column 2, line 36; figure 1 *	1-6	G 06 F 5/06
A	US-A-4 394 753 (PENZEL) * Claims 1,2; figure 1 *	1	

TECHNICAL FIELDS SEARCHED (Int. Cl.5)

G 06 F
G 11 C

The present search report has been drawn up for all claims

Place of search	Date of completion of search	Examiner
The Hague	19 February 91	COHEN B.

CATEGORY OF CITED DOCUMENTS

X: particularly relevant if taken alone
Y: particularly relevant if combined with another document of the same category
A: technological background
O: non-written disclosure
P: intermediate document
T: theory or principle underlying the invention

E: earlier patent document, but published on, or after the filing date
D: document cited in the application
L: document cited for other reasons

&: member of the same patent family, corresponding document